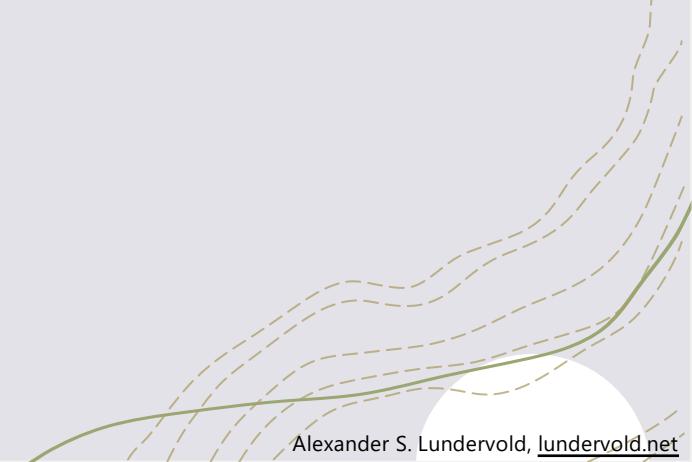


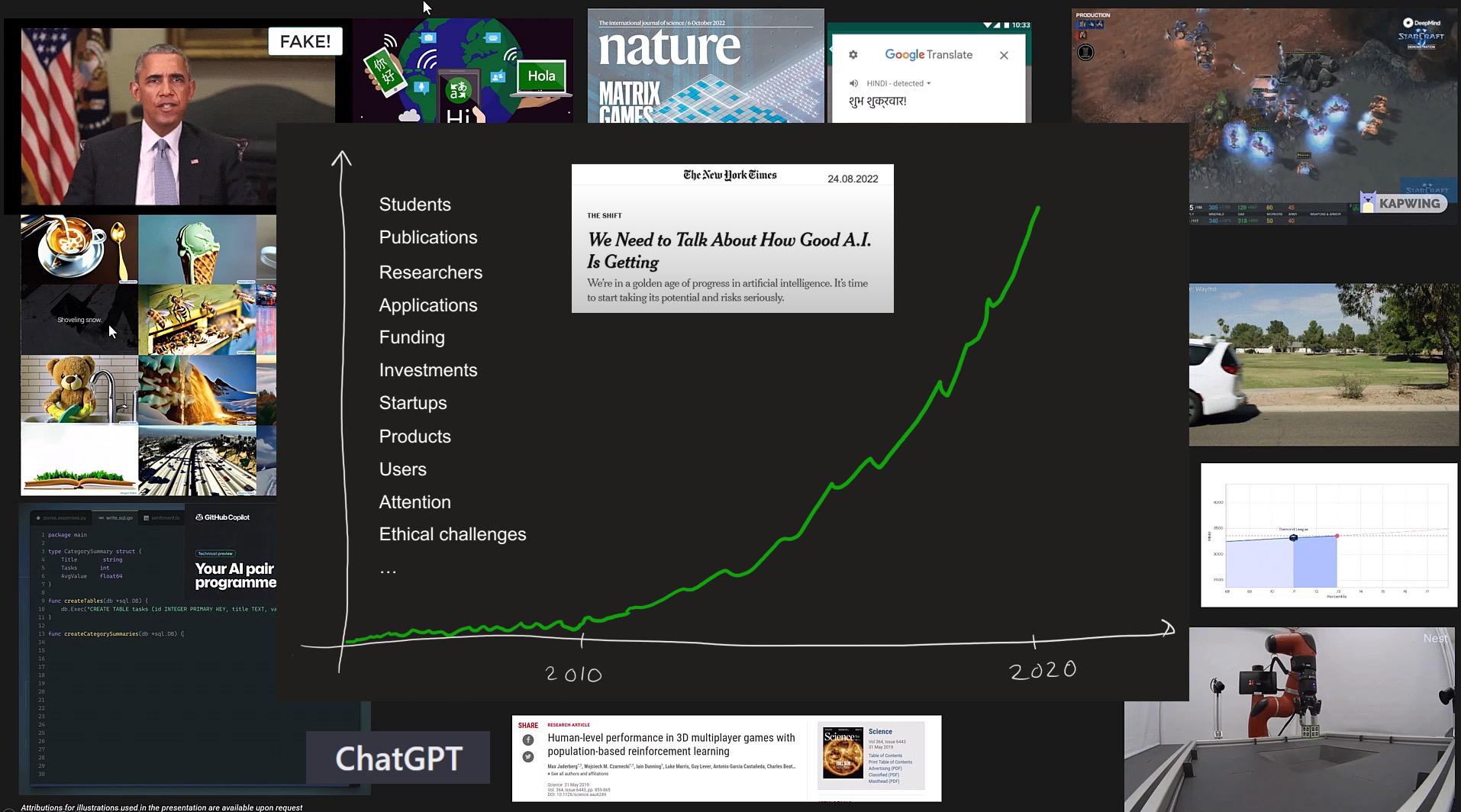


# Lab 3: Deep learning

ELMED219 – 2023

Friday, January 20<sup>th</sup>, 2023





# Plan



Hands-on



What *is* deep  
learning?



How do you *do*  
deep learning?



Recent  
developments in  
generative AI

## Deep learning

*searching for good hierarchical geometric representations*

## Deep learning

***searching for good hierarchical geometric representations***

## Deep learning

*searching for **good** hierarchical geometric representations*

**Deep learning**

*searching for good **hierarchical geometric representations***

**Deep learning**

*searching for good hierarchical geometric representations*

01

02

03

04

05

Function approximation

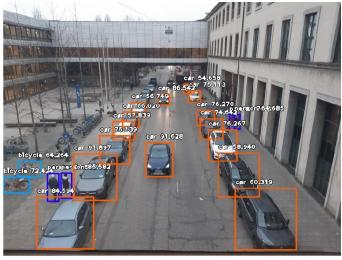
Function approximation

$$y \approx f(x; \theta)$$

what's in the image

an image

The diagram illustrates the concept of function approximation. It features a mathematical equation  $y \approx f(x; \theta)$ . Two arrows point from descriptive text to specific parts of the equation: one arrow points from the phrase "what's in the image" to the input variable  $x$ , and another arrow points from the phrase "an image" to the output variable  $y$ .



## Function approximation

$$y \approx f(x; \theta)$$

what's in the image

an image

Arrows point from the text "what's in the image" to the variable  $x$  in the equation, and from the text "an image" to the variable  $\theta$ .

Function approximation

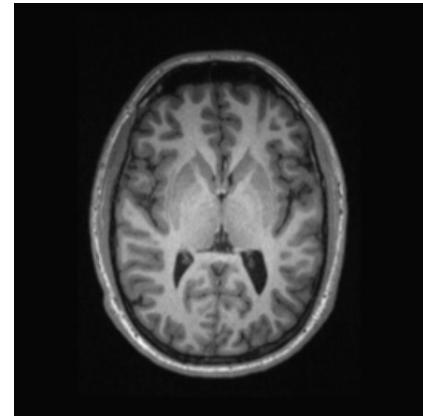
$$y \approx f(x; \theta)$$

what's in the image

an image

The diagram illustrates the concept of function approximation. It features a mathematical equation  $y \approx f(x; \theta)$  centered on the page. Two arrows originate from descriptive text at the bottom: one points to the input variable  $x$  in the equation, and the other points to the output variable  $y$ . The text 'what's in the image' is positioned near the bottom left, and the text 'an image' is positioned near the bottom right.

AGE



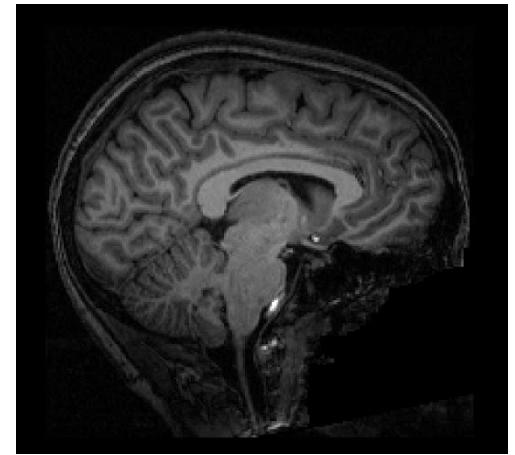
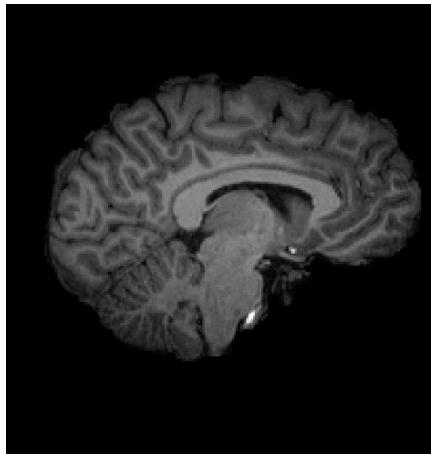
Function approximation

$$y \approx f(x; \theta)$$

an MRI image

"Brain age"

A mathematical equation  $y \approx f(x; \theta)$  is shown. An arrow points from the term "Brain age" to the variable  $x$  in the equation. Another arrow points from the text "an MRI image" to the image of the brain above it.



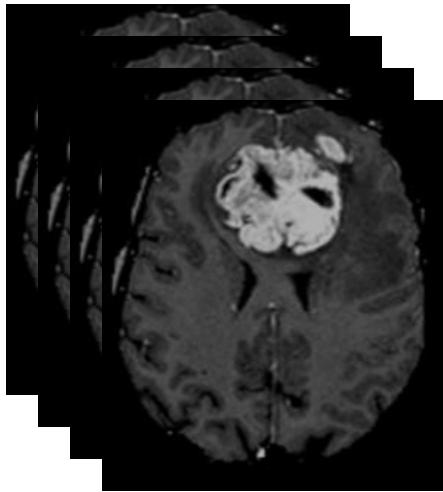
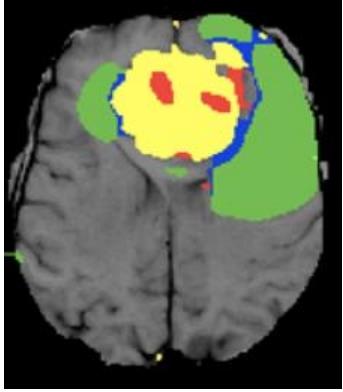
Function approximation

$$y \approx f(x; \theta)$$

skull-stripped MRI

a brain MRI

Two arrows point from the labels "skull-stripped MRI" and "a brain MRI" to the corresponding terms in the function approximation equation  $y \approx f(x; \theta)$ .



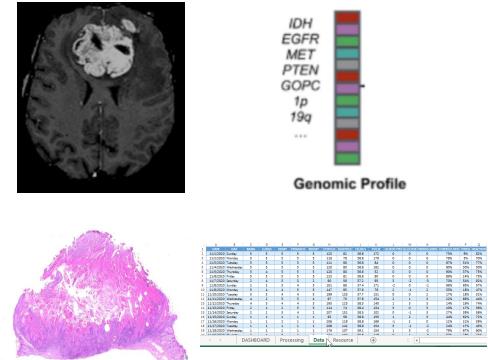
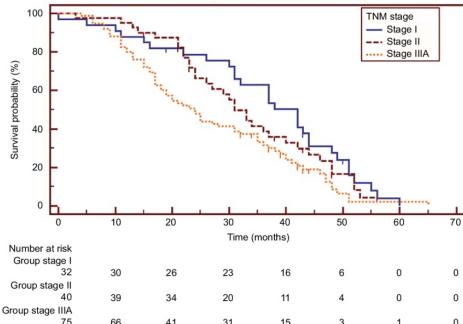
Function approximation

$$y \approx f(x; \theta)$$

multimodal MRI images

Segmentation of region-of-interests (e.g., tumors)

Two arrows point from the text labels "multimodal MRI images" and "Segmentation of region-of-interests (e.g., tumors)" to the corresponding images above.



## Function approximation

$$y \approx f(x; \theta)$$

outcome (e.g., survival)

heterogeneous information

Text-to-speech  
Speech-to-speech  
Text-to-image  
Time-series to anomaly yes/no  
and much more...

Function approximation

$$y \approx f(x; \theta)$$

Training

$$(X, y)$$

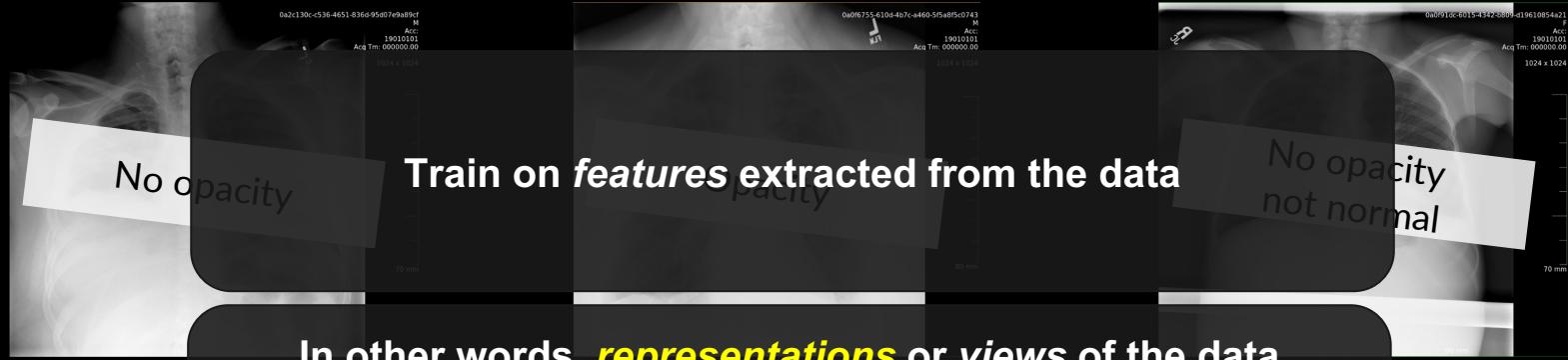
## Training

$$(X, y)$$

## Input data

## Labels

*This is the setup in what's called **supervised learning***



In other words, **representations** or views of the data

*Important that these representations are sufficiently informative*



$$y \approx f(x ; \theta)$$

# 01 Machine learning: function approximation based on training

$$y \approx f(x ; \theta)$$

features of a disease,  
process, patient or  
system

healthy or not healthy /  
outcome

The diagram illustrates a machine learning function approximation model. At the center is the equation  $y \approx f(x ; \theta)$ . A yellow arrow points from the left towards the output variable  $y$ , which is labeled "healthy or not healthy / outcome". Another yellow arrow points from the right towards the input variable  $x$ , which is labeled "features of a disease, process, patient or system".

## Models and parameters

$$y \approx f(x; \theta)$$


The diagram consists of two arrows originating from the words "model" and "parameters" at the bottom left and right respectively, and pointing upwards towards the function  $f(x; \theta)$ .

**Training** is a search for useful representations through a space defined by a family  $\mathcal{F}$  of parametrized models

**01** Machine learning: **function approximation** based on **training**

**02** We pick the model family  $F$ . The parameters  $\Theta$  are found automatically through training

$$y \approx f(x; \theta)$$

## Theory versus practice

Theory       $X \longrightarrow Y$

**In theory:**

can get arbitrarily good approximations by training simple models  
(*universal approximation* and *no free lunch theorems*)

Practice       $X \xrightarrow{\text{complex function}} Y$

**In practice:**

which  $f$  we use and how the training is done makes a huge difference

The family  $F$  of models considered, i.e., the *hypothesis space*, should be

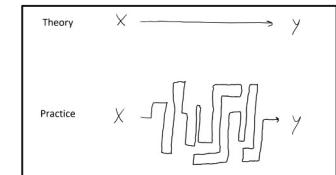
- (i) **efficiently searchable**,
- (ii) leading to **expressive** models that can
- (iii) **generalize** beyond the training data distribution (i.e., work well on new data)

**01** Machine learning: **function approximation** based on **training**

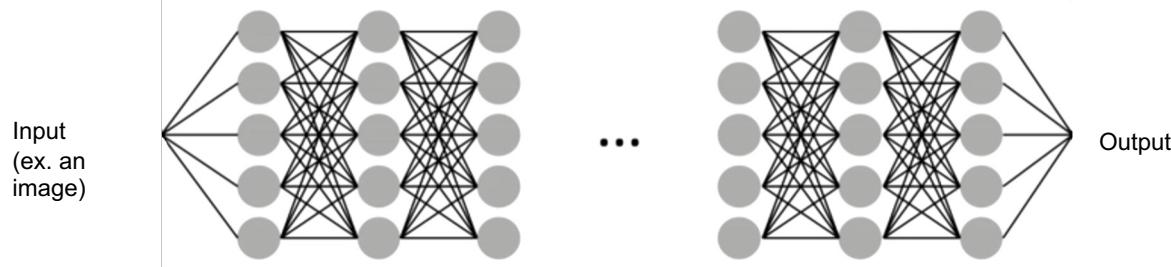
**02** We pick the model family  $F$ . The parameters  $\Theta$  are found automatically through training

**03** In practice: some models are better suited to some tasks than others.

$$y \approx f(x; \theta)$$

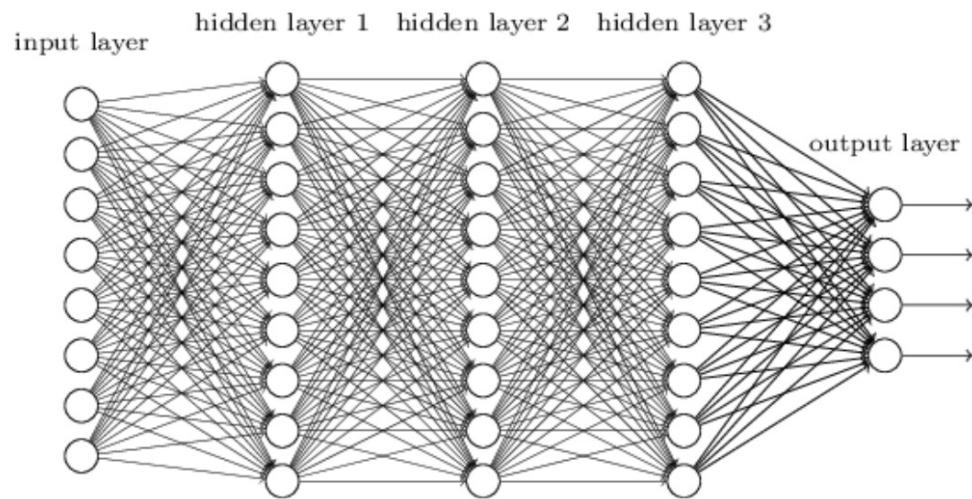


$$y \approx f(x; \theta)$$

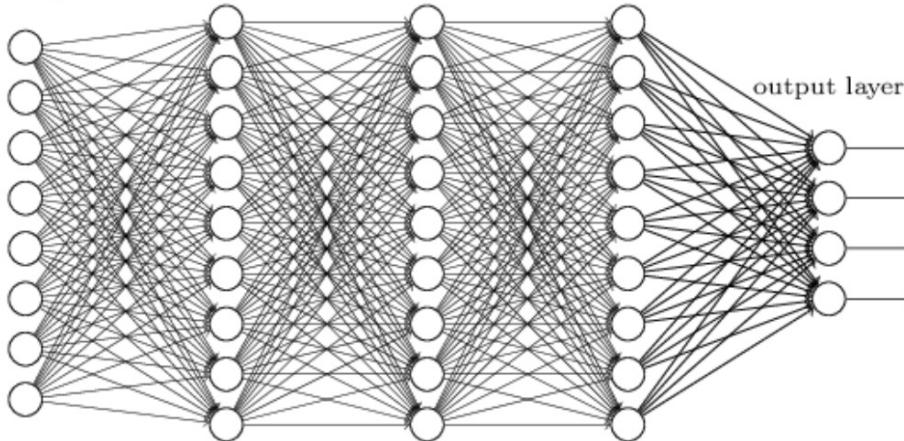


### Artificial neural networks

Computational graphs of simple units (“neurons”) connected in various specific ways, parametrized by parameters associated to each layer.



input layer      hidden layer 1    hidden layer 2    hidden layer 3



A geometric analogy:  
uncrumpling paper



This is a fancy way to draw what's really just a composition of simple functions:

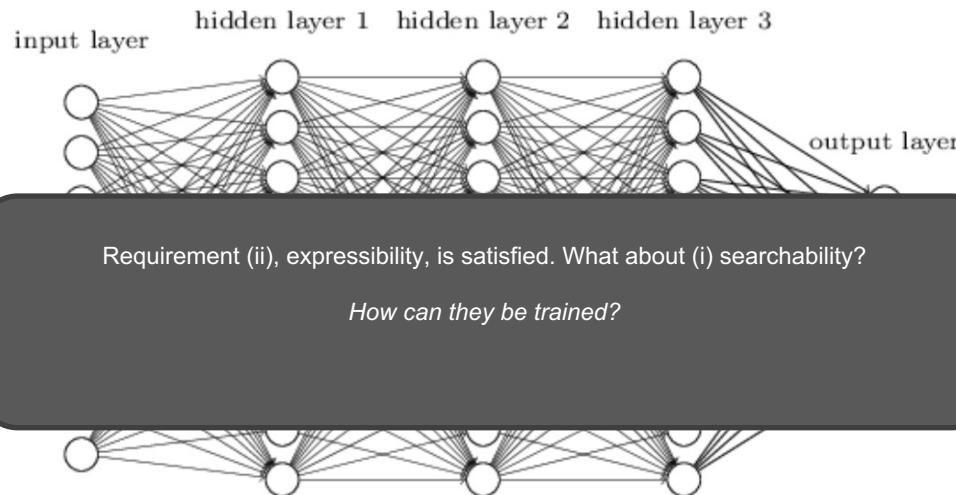
$$y = \sigma_{k+1}(W_{k+1}^T \sigma_k(W_k^T \left( \dots \left( \sigma_1(W_1^T x) \right) \dots \right)) = f(x; \theta)$$

Input  $x$  is sent through several consecutive **layers** (functions). An output is produced.

Think of each layer as a (geometric) **transformation** of its input.

Every layer has a bunch of **parameters**, stored in the matrices  $W$

*Each family member of  $F$  is complex transformation broken down into a composition of simpler ones, parametrized by the layer weights.*



This is a fancy way to draw what's really just a composition of simple functions:

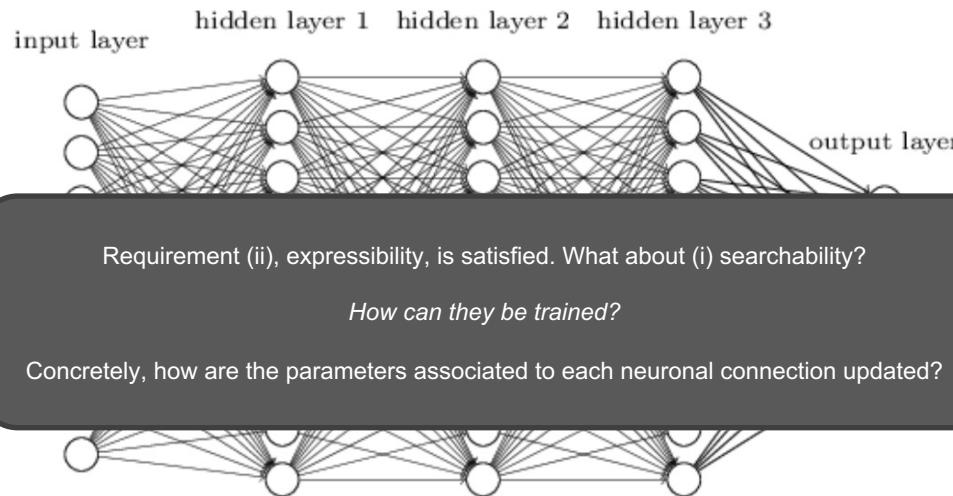
$$y = \sigma_{k+1}(W_{k+1}^T \sigma_k(W_k^T \left( \dots \left( \sigma_1(W_1^T x) \right) \dots \right)) = f(x; \theta)$$

Input  $x$  is sent through several consecutive **layers** (functions). An output is produced.

Think of each layer as a (geometric) **transformation** of its input.

Every layer has a bunch of **parameters**, stored in the matrices  $W$

*Each family member of  $F$  is complex transformation broken down into a composition of simpler ones, parametrized by the layer weights.*



This is a fancy way to draw what's really just a composition of simple functions:

$$y = \sigma_{k+1}(W_{k+1}^T \sigma_k(W_k^T \left( \dots \left( \sigma_1(W_1^T x) \right) \dots \right)) = f(x; \theta)$$

Input  $x$  is sent through several consecutive **layers** (functions). An output is produced.

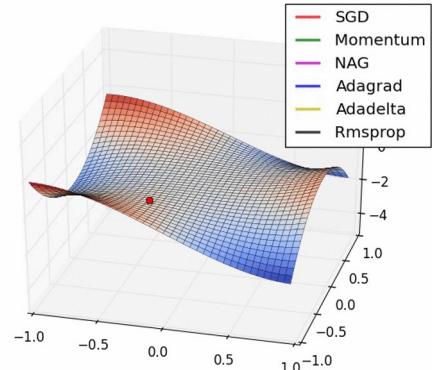
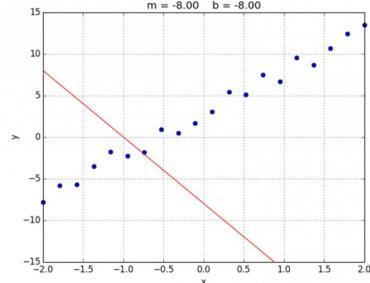
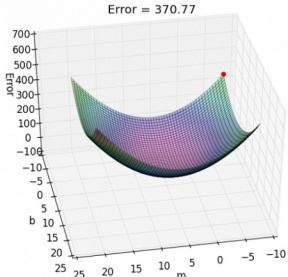
Think of each layer as a (geometric) **transformation** of its input.

Every layer has a bunch of **parameters**, stored in the matrices  $W$

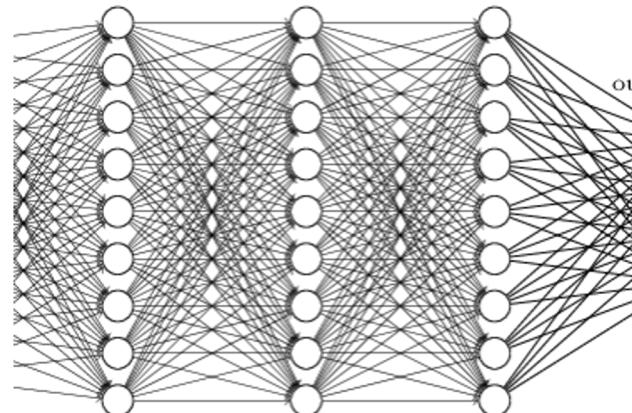
*Each family member of  $F$  is complex transformation broken down into a composition of simpler ones, parametrized by the layer weights.*

# Three basic ingredients:

1. A **loss function**: Used to measure the network's performance on the training data. A **feedback signal** for the training process.
2. An **optimizer**: Used to update the *parameters* of the network to increase performance as measured by the loss function.  
**Gradient descent and backpropagation.**
3. One or more **metrics**: A way to score the model. For example *accuracy* or *mean squared error*.



hidden layer 1   hidden layer 2   hidden layer 3



Define a neural network model with an architecture and a set of trainable parameters (weights and biases)

1. Initialize the parameters

2. Collect a batch of inputs ( $X, y$ ) and use the model with its current parameters to make predictions  $y_{pred}$

3. Calculate how good the model is using the loss function to measure the discrepancy between the known labels  $y_{true}$  and the predictions  $y_{pred}$  for this batch

4. Calculate the gradients for each parameter using backpropagation. This measures how changing each parameter would change the loss. In other words, each parameter's contribution to the loss.

5. Move each weight a little bit in the opposite direction of its gradient using gradient descent. i.e., some variant of  
$$\text{parameter} = \text{parameter} - \text{learning\_rate} * \text{gradient}$$

6. Go back to step 2 and repeat the process

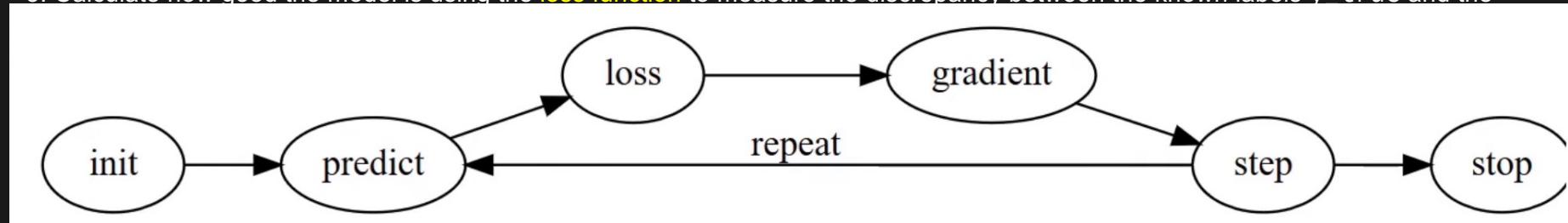
7. Do this over and over for several epochs (one epoch is one pass through all the training data), until you meet some chosen stopping criteria

Define a neural network model with an architecture and a set of trainable parameters (weights and biases)

1. Initialize the parameters

2. Collect a batch of inputs ( $X, y$ ) and use the model with its current parameters to make predictions  $y_{pred}$

3. Calculate how good the model is using the loss function to measure the discrepancy between the known labels  $y_{true}$  and the



5. Move each weight a little bit in the opposite direction of its gradient using gradient descent. I.e., some variant of  
$$\text{parameter} = \text{parameter} - \text{learning\_rate} * \text{gradient}$$

6. Go back to step 2 and repeat the process

7. Do this over and over for several epochs (one epoch is one pass through all the training data), until you meet some chosen stopping criteria

Define a neural network model with an architecture and a set of trainable parameters (weights and biases)

1. Initialize the parameters

2. Collect a batch of inputs ( $X, y$ ) and use the model with its current parameters to make predictions  $y_{pred}$

3. Calculate how good the model is using the loss function to measure the discrepancy between the known labels  $y_{true}$  and the predictions  $y_{pred}$  for this batch

## What is gradient descent?

4. Calculate the gradients for each parameter. This means calculating how much changing each parameter would change the loss. In other words, each parameter's contribution to the loss.

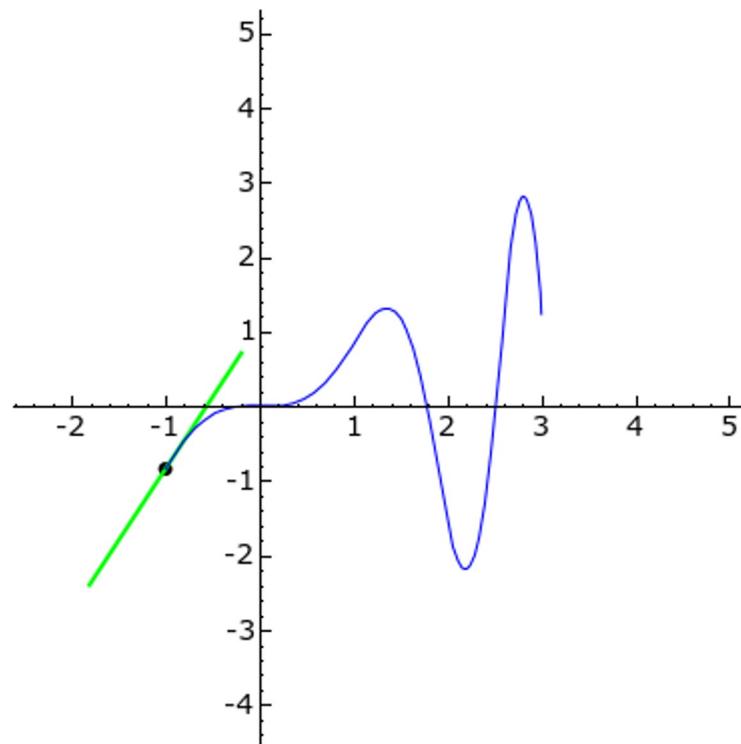
5. Move each weight a little bit in the opposite direction of its gradient using gradient descent. i.e., some variant of  
$$\text{parameter} = \text{parameter} - \text{learning\_rate} * \text{gradient}$$

6. Go back to step 2 and repeat the process

7. Do this over and over for several epochs (one epoch is one pass through all the training data), until you meet some chosen stopping criteria

Define a neural network model with an architecture and a set of trainable parameters (weights and biases)

1. Init



2. Co

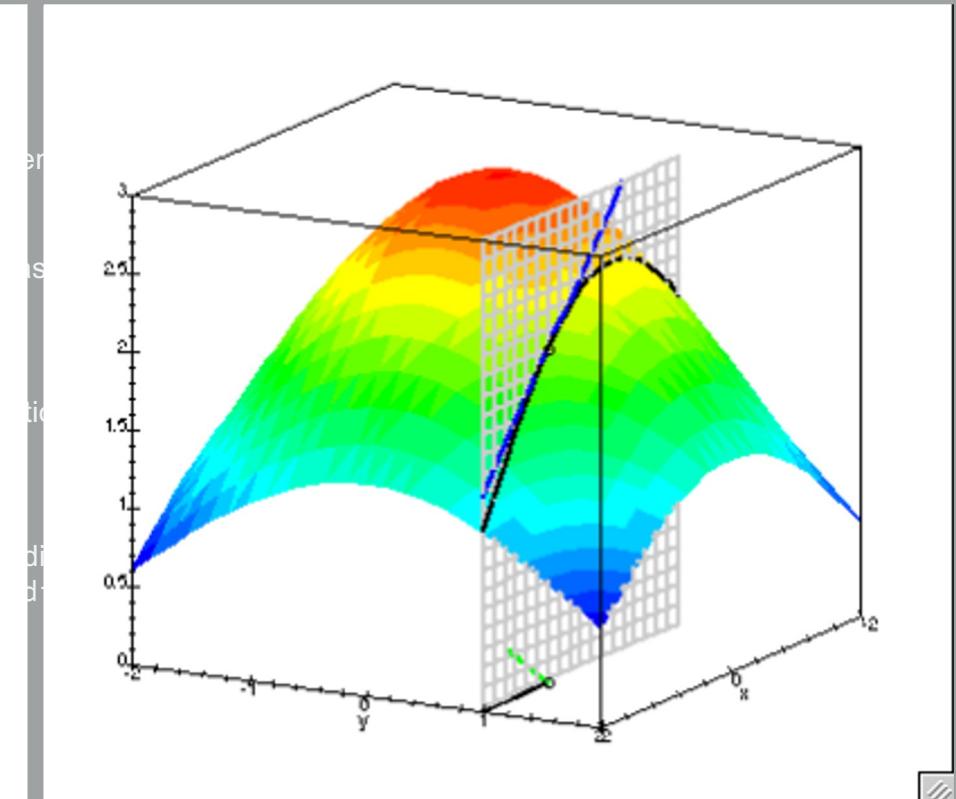
3. Ca

4. Ca

5. M

6. Go

7. Do this over and over for several epochs (one epoch is one pass through all the training data), until you meet some chosen stopping criteria



Define a neural network model with an architecture and a set of trainable parameters (weights and biases)

1. Init

2. Co

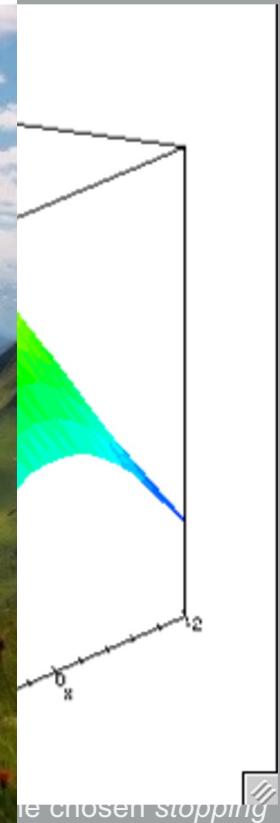
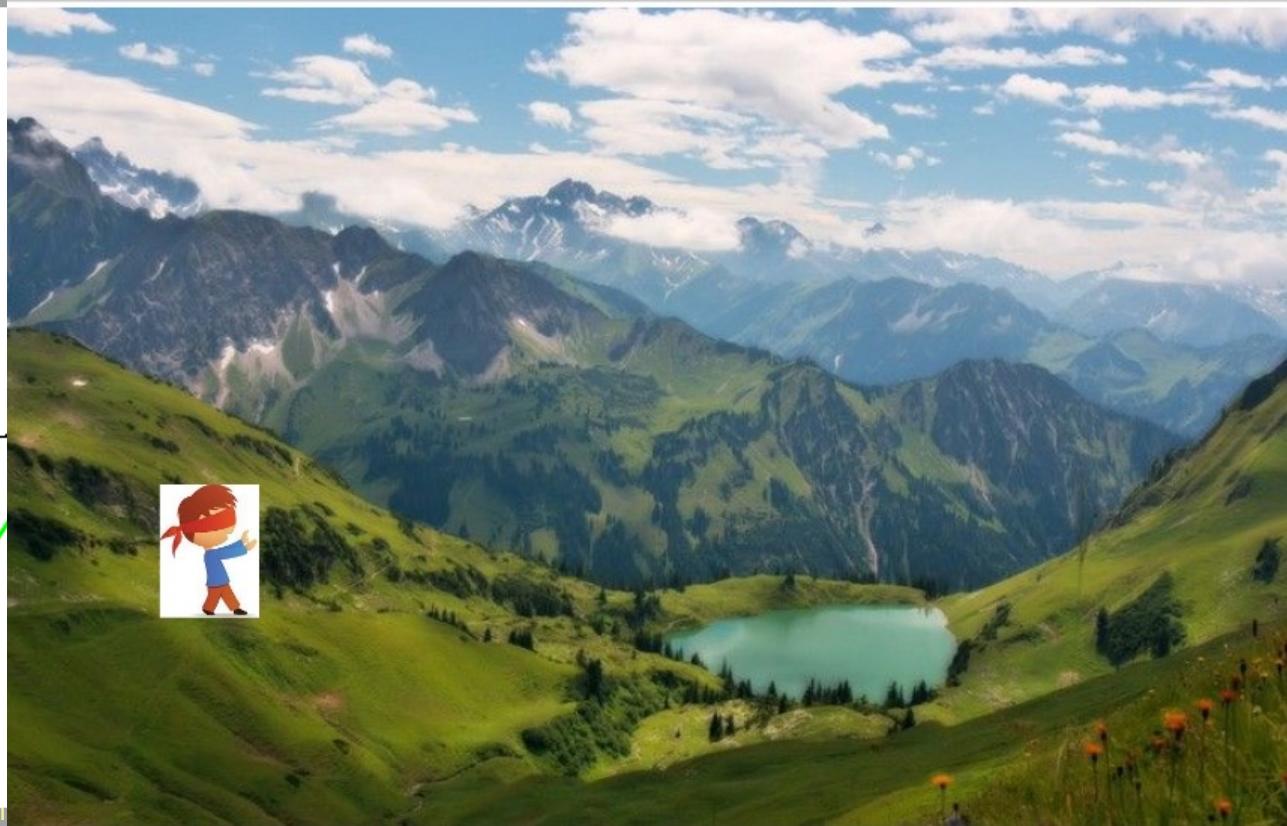
3. Ca  
predi

4. Ca  
loss.

5. M

6. Go

7. Do this over an  
criteria



the chosen stopping

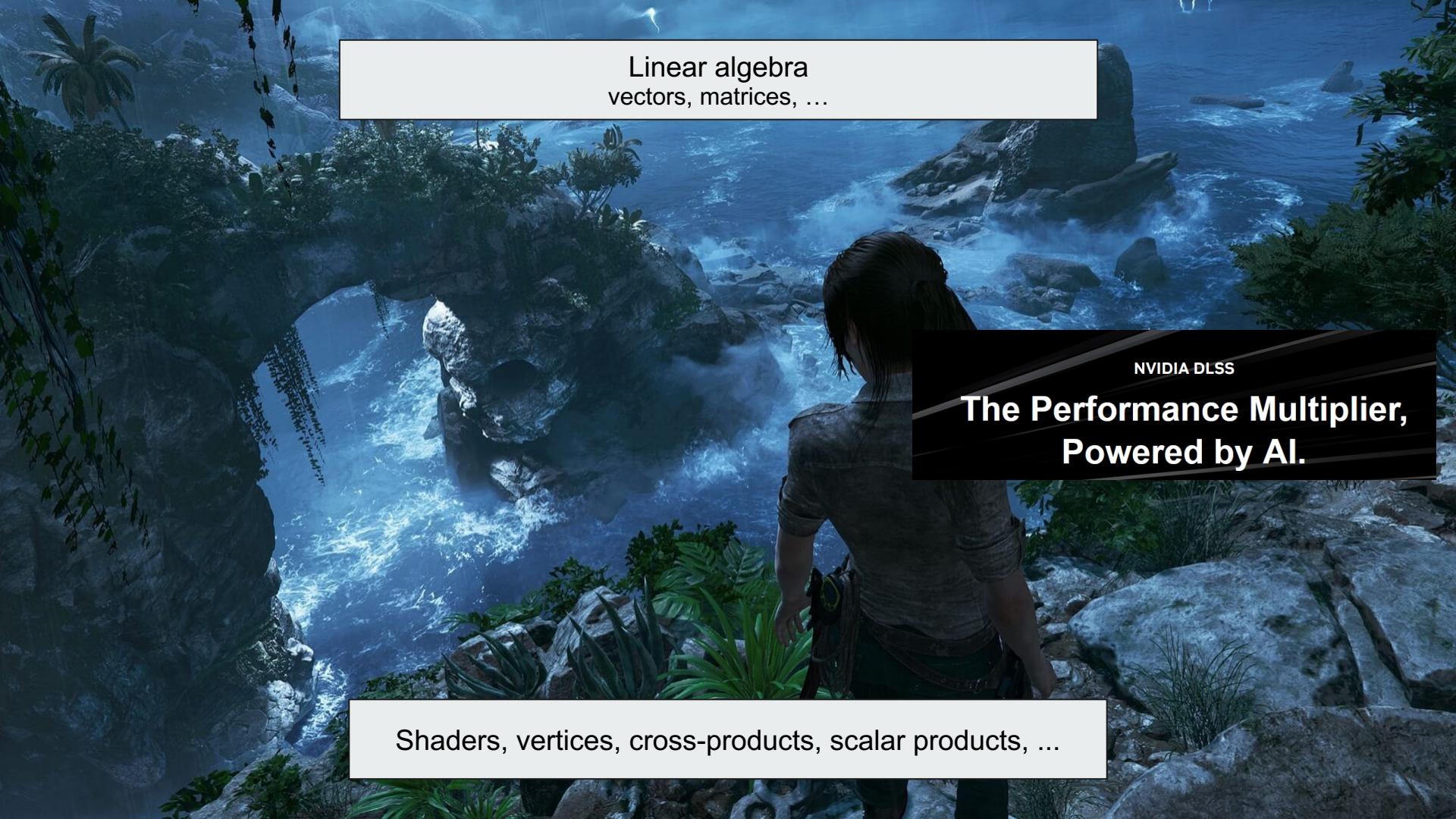


GPUs: *Massively parallel linear algebra super-computers*

Developed for computer graphics

*Good at exactly what's needed for neural networks!*





Linear algebra  
vectors, matrices, ...

NVIDIA DLSS

**The Performance Multiplier,  
Powered by AI.**

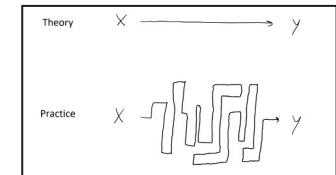
Shaders, vertices, cross-products, scalar products, ...

**01** Machine learning: **function approximation** based on **training**

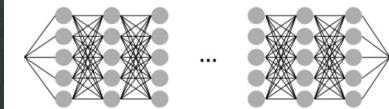
$$y \approx f(x; \theta)$$

**02** We pick the model family  $F$ . The parameters  $\Theta$  are found automatically through training

**03** In practice: some models are better suited to some tasks than others.

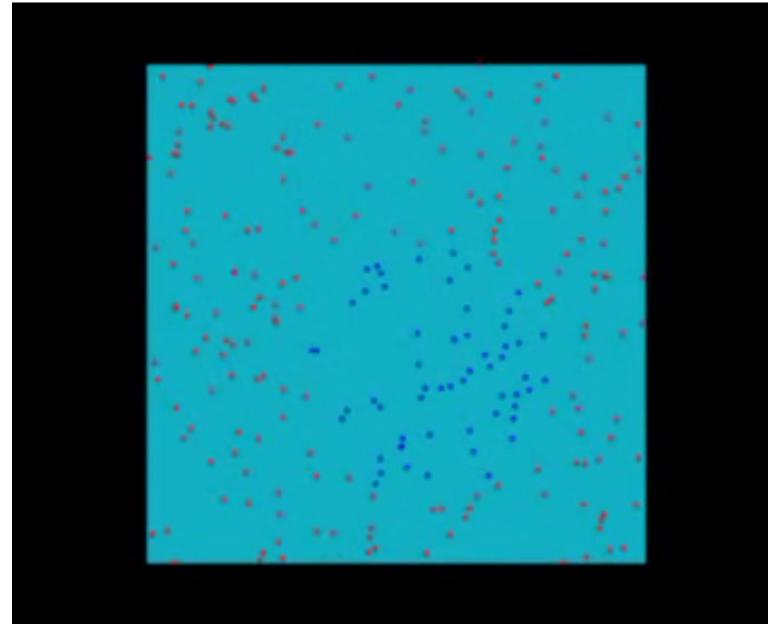


**04** In deep learning: a particular choice of  $F$ . A **composition** of simple functions, trained jointly, typically using GPUs or other accelerators.

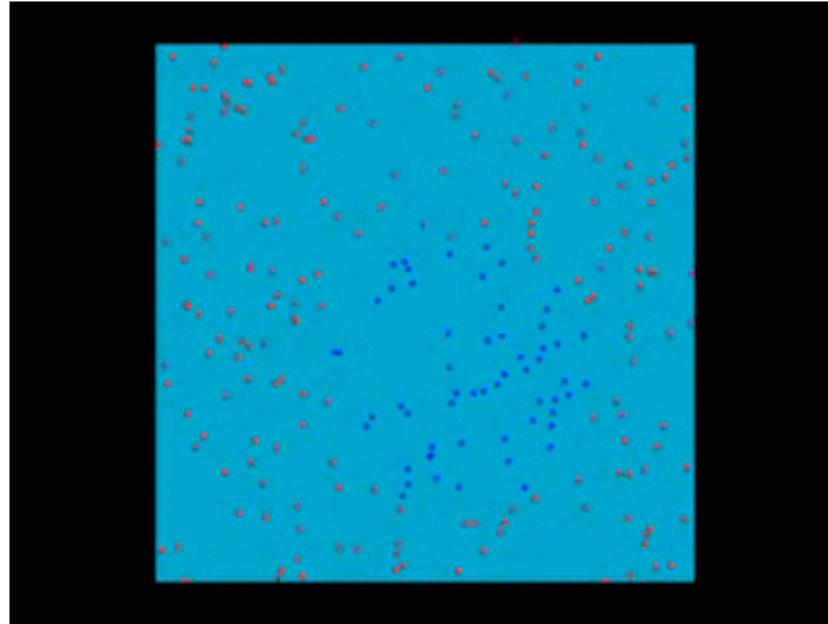


Representations and the power of transformations

## Transformations

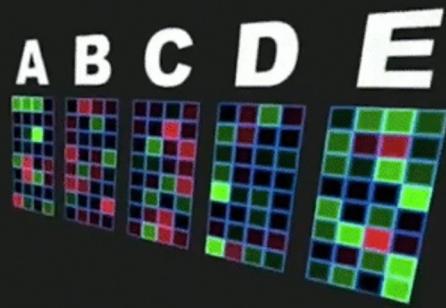


## Transformation

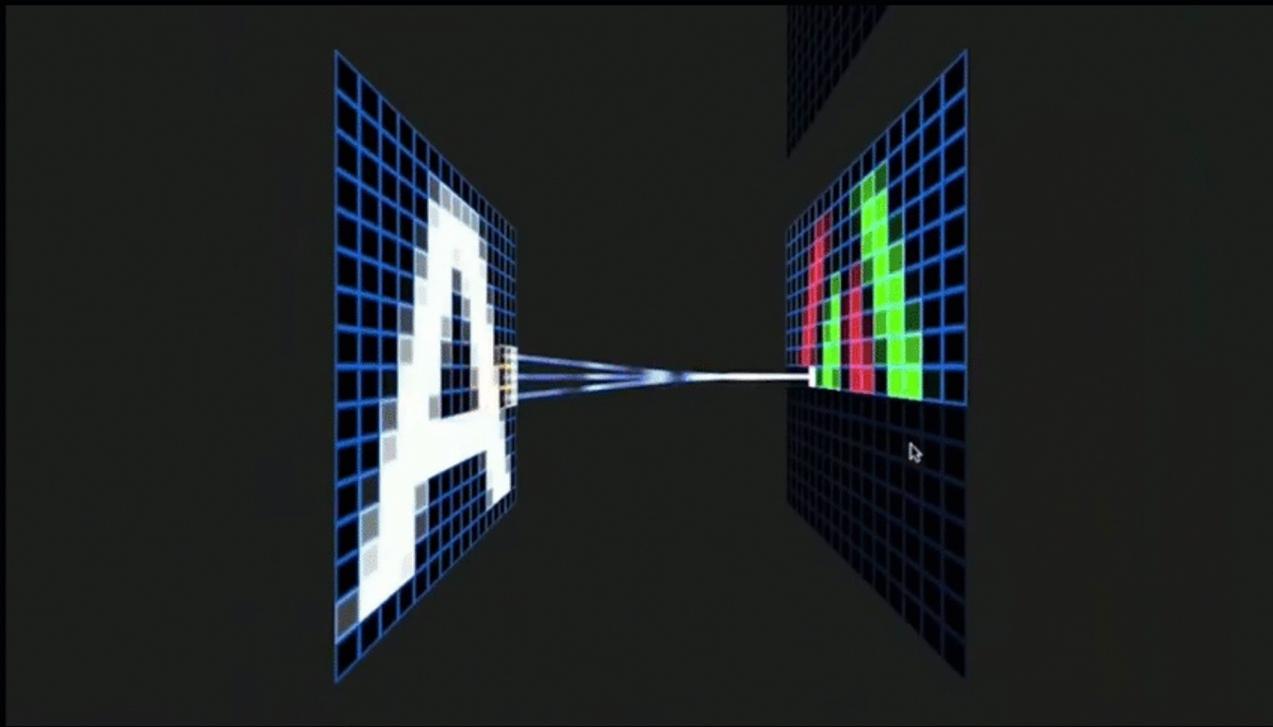


What transformations should be applied?

In deep learning: ***transformations providing useful representations automatically found via training***



↓



**01**

Machine learning: **function approximation** based on **training**

**02**

We pick the model family  $F$ . The parameters  $\Theta$  are found automatically through training

**03**

In practice: some models are better suited to some tasks than others.

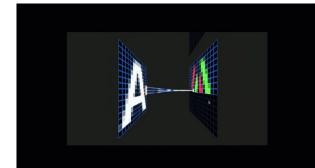
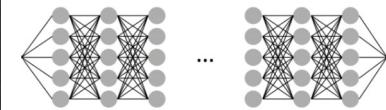
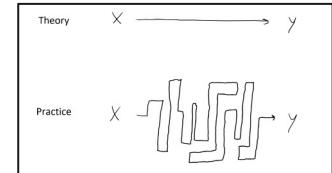
**04**

In deep learning: a particular choice of  $F$ . A **composition** of simple functions, trained jointly, typically using GPUs or other accelerators.

**05**

In deep learning: finding a **hierarchical set of geometric transformations**, ending in a representation that makes the desired task easy to solve.

$$y \approx f(x; \theta)$$



**01**

Machine learning: **function approximation** based on **training**

**02**

We pick the model family  $F$ . The parameters  $\Theta$  are found automatically through training

**03**

In practice: some models are better suited to some tasks than others.

### Deep learning

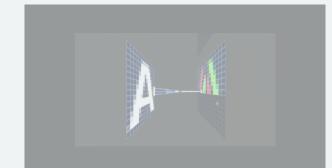
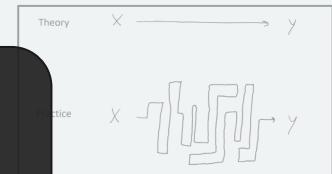
**04**

*searching for good hierarchical geometric representations*  
In deep learning: a particular choice of  $F$ . A composition of simple functions, trained jointly, typically using GPUs or other accelerators.

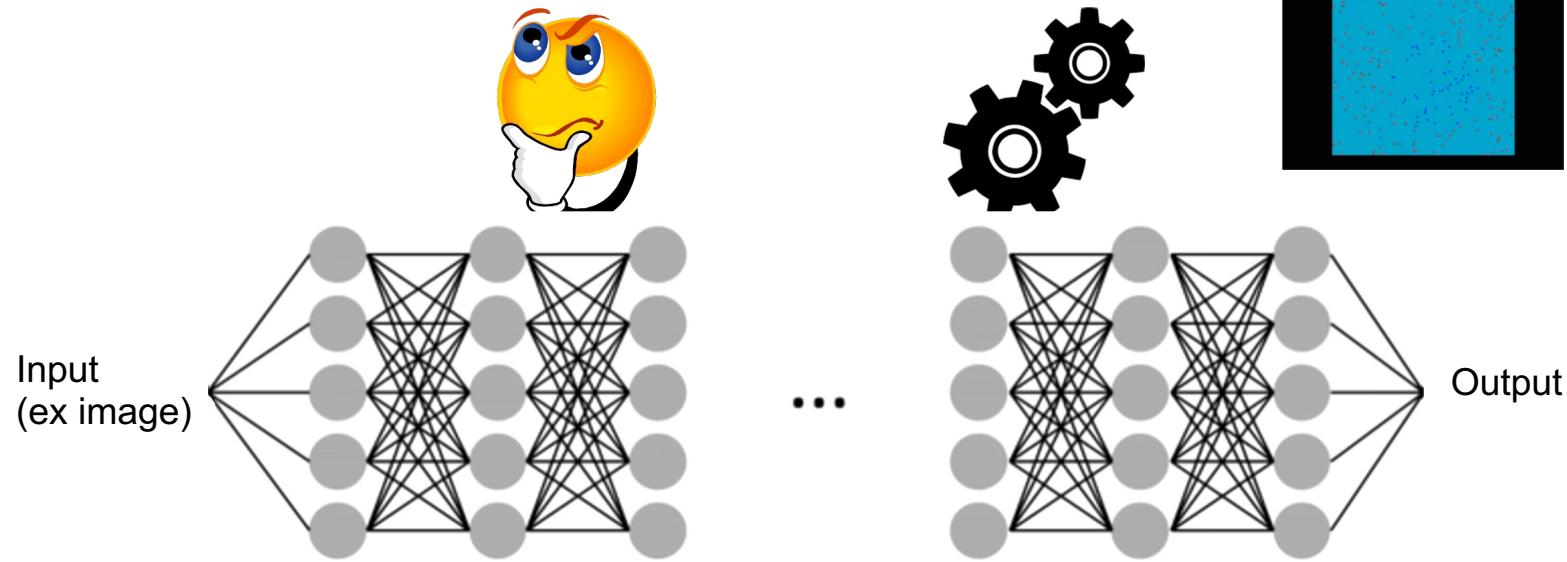
**05**

In deep learning: finding a **hierarchical set of geometric transformations**, ending in a representation that makes the desired task easy to solve.

$$y \approx f(x; \theta)$$



What distinguishes deep learning from other kinds of machine learning?

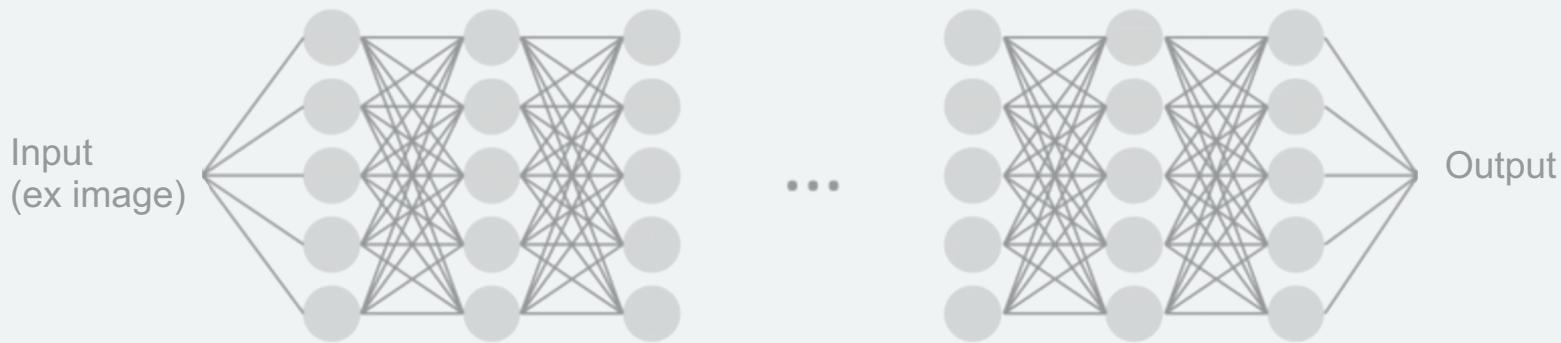


“**Classical** machine learning  
Artificial neural network

Deep learning

**In practice, this is a bit of a lie**

(albeit a useful lie, somewhat true)



**Deep learning**  
Artificial neural network

# Plan



What *is* deep  
learning?



How do you *do*  
deep learning?



Recent  
developments in  
generative AI

jupyter 1.0-asi-brain\_tumor\_analysis\_radiomics Last Checkpoint: an hour ago Junusseit changed Logout

A.S. Lundervold, 12.09.22

### Using deep learning to extract imaging-biomarkers for brain cancer analyses in MRI of glioblastoma



In this notebook, we'll use a deep learning model to segment brain tumors from multi-parametric MRI and then extract features from the resulting tumor masks. Such features can potentially be associated with tumor severity and prognosis and contribute to better treatment.

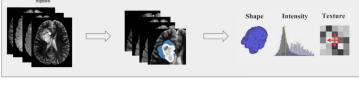
Extracting features from objects of interest in medical images for diagnostic purposes is often referred to as radiomics. The goal of radiomics is to extract information from medical images that can be used for predicting clinical outcomes. Radiomic features are often extracted from various imaging modalities, e.g., different MRI sequences, X-ray imaging, CT imaging, and so on. One can then combine it with other sources of information such as demographic, clinical data, genetics. In such a way, radiomics and radiogenomics can open the door to sophisticated and powerful analyses.

#### Radiomics workflow

In this notebook, we will estimate the location and extent of the brain tumor, TD-enhancing and non-enhancing regions. We will then use this to extract the tumor location and the tumor number. Additionally, we can look at the features of the MRI images inside each of these two tumor parts.

```

graph LR
    A[Image acquisition] --> B[Segmentation]
    B --> C[Feature extraction]
    C --> D[Shape]
    C --> E[Intensity]
    C --> F[Texture]
  
```



jupyter 2.0-asi-deep\_learning\_brain\_tumor\_segmentation Last Checkpoint: Yesterday at 11:54 AM (auto-saved) Logout

Alexander S. Lundervold, 07.09.22

### Introduction

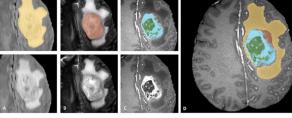
This notebook is based on the BrainX segmenter (an Jupyter notebook from <https://github.com/Pheon/MONAI>). You should consult the MONAI documentation for a more thorough introduction to the MONAI framework: <https://monai.io>.

The notebook will give a hands-on example of deep learning in practice. Along the way, we will see some of the deep learning concepts covered in the introduction to the workshop translated into code.

Our goal is to illustrate a possible approach to brain tumor segmentation, i.e., extracting meaningful tumor sub-regions. Having an accurate brain tumor segmentation is important for many downstream applications such as target volumes and locations, and also provides regions of interest for extracting radiomic features. In another notebook, we take a practical look at radiomics.

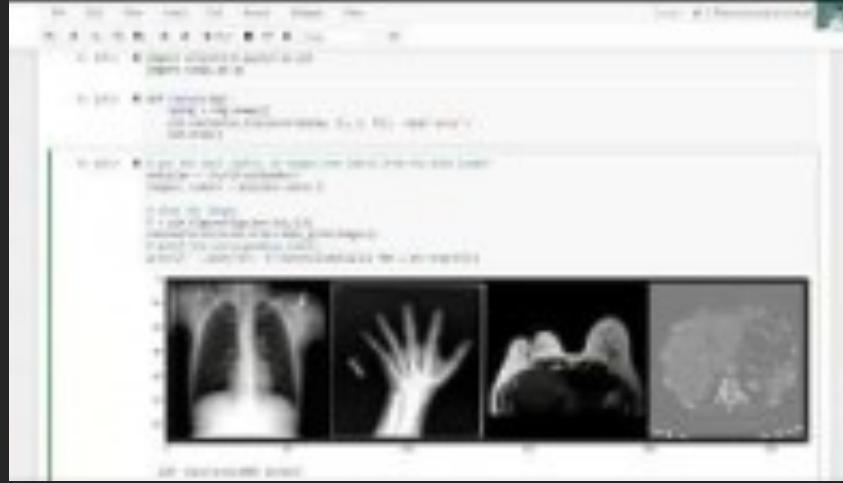
### The data: The Brain Tumor Segmentation (BraTS) challenge

We will use the data from the BraTS 2016 and 2017 competitions, consisting of 700 multi-parametric MRI studies of patients with brain tumors. Each study consists of four MRI modalities, T1w, contrast-enhanced T1w, T2w, and FLAIR, and corresponding manual delineations of three tumor regions: necrotic tumor, active tumor, and edema.



### Sample images

It is a heterogeneous data set. Here are some sample images:

[fastMONAI](#)

Overview  
Vision core plot  
Vision core  
Vision data  
Data augmentation  
Custom loss functions  
Vision metrics  
Utils  
Dataset information  
External data  
Tutorials  
Single-label classification  
Regression  
Binary semantic segmentation  
Multi-class semantic segmentation

On this page  
[Installing](#)  
[From PyPI](#)  
[From GitHub](#)  
[Getting started](#)  
[How to contribute](#)  
[Citing fastMONAI](#)

A low-code Python-based open source deep learning library built on top of [fastai](#), [MONAI](#), and [TorchIO](#).

fastMONAI simplifies the use of state-of-the-art deep learning techniques in 3D medical image analysis for solving classification, regression, and segmentation tasks. fastMONAI provides the users with functionalities to step through data loading, preprocessing, training, and result interpretations.

Note: This documentation is also available as interactive notebooks.

## Installing

# Plan



What *is* deep  
learning?



How do you *do*  
deep learning?



Recent  
developments in  
generative AI



# Generative AI

Curated by @aaronsim

DALL-E 2 [Stable Diffusion](#) [craiyon](#) [Lexica](#) [MidJourney](#)  
Imagen [Wombo](#) [NightCafe](#) [GauGAN2](#) [DeepAI](#) [Jasper](#) [Artbreeder](#)  
Wonder [picrav-text2image](#) [neural](#) [love](#) [Omneky](#) [alpaca](#)  
[image.sphere](#) [KREA](#) [Nyx+ gallery](#) [ROSEBUD.AI](#) [PhotoRoom](#)

Text-to-Image (T2I)

[runway](#) [Flikr](#) [synthesia](#) [Meta AI](#) [Google AI](#) [Phenaki](#) [CONTENDA](#)

Text-to-Audio (T2A)

[Play.ht](#) [MURF](#) [RESEMBLE.AI](#) [WELLSAID](#)  
[Simplified](#) [Jasper](#) [Frase](#) [LectorNet](#)  
[InterKit](#) [GooseAI](#) [MarketMus](#) [AI21 Labs](#) [HubSpot](#)  
[grammarly copy.ai](#) [neural](#) [love](#)  
[ideasbyai](#) [textortex](#) [OpenAI GPT-3](#) [Blog Ideas](#)  
[HyperText](#) [Subtxt](#) [WRITER](#) [YouTube](#) [LAIKA](#)  
[Moobeam](#) [BerthaAI](#) [anyword](#) [Hypoten](#)

Text-to-Motion (T2M)

[TREE.net](#) MDM Human Motion Diffusion Model

Text-to-Code (T2C)

[replit](#) [Ghostwriter](#) [GitHub Copilot](#) [MUTABLE](#)  
[Amazon CodeWhisperer](#)

Text-to-NFT (T2N)

[LensAI](#)

DreamFusion [CLIP-Mesh](#) [GET3D](#)

Audio-to-Text (A2T)

[descript](#) [AssemblyAI](#) [Whisper](#)

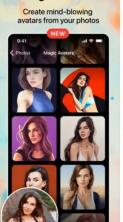
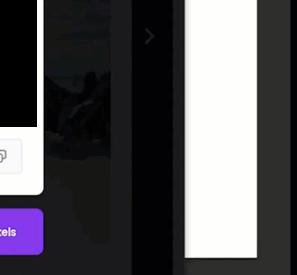
AudioLM [NN-VOICEMOD](#)

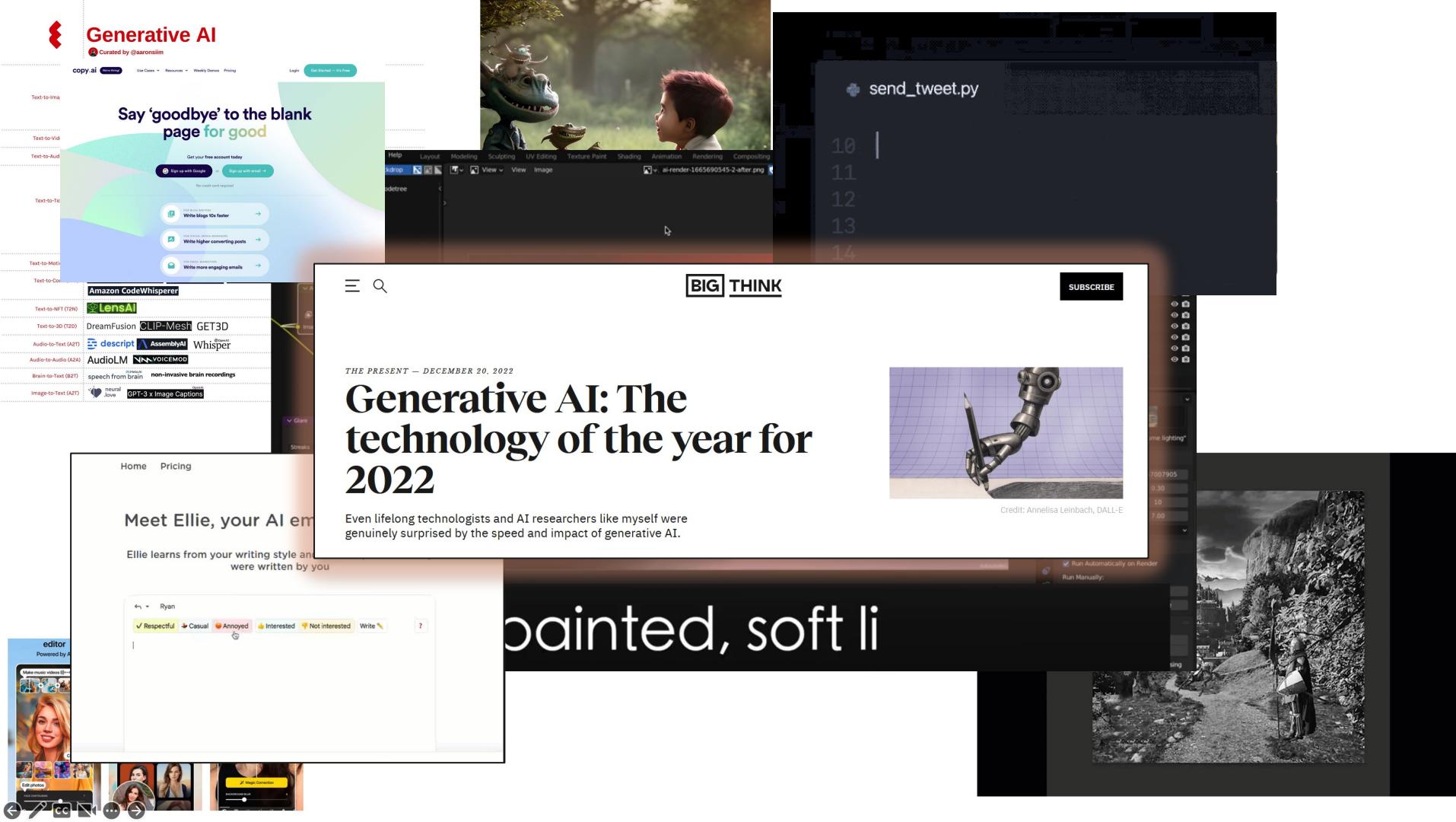
Brain-to-Text (B2T)

speech from brain non-invasive brain recordings

Image-to-Text (A2I)

[neural](#) [love](#) [GPT-3 x Image Captions](#)



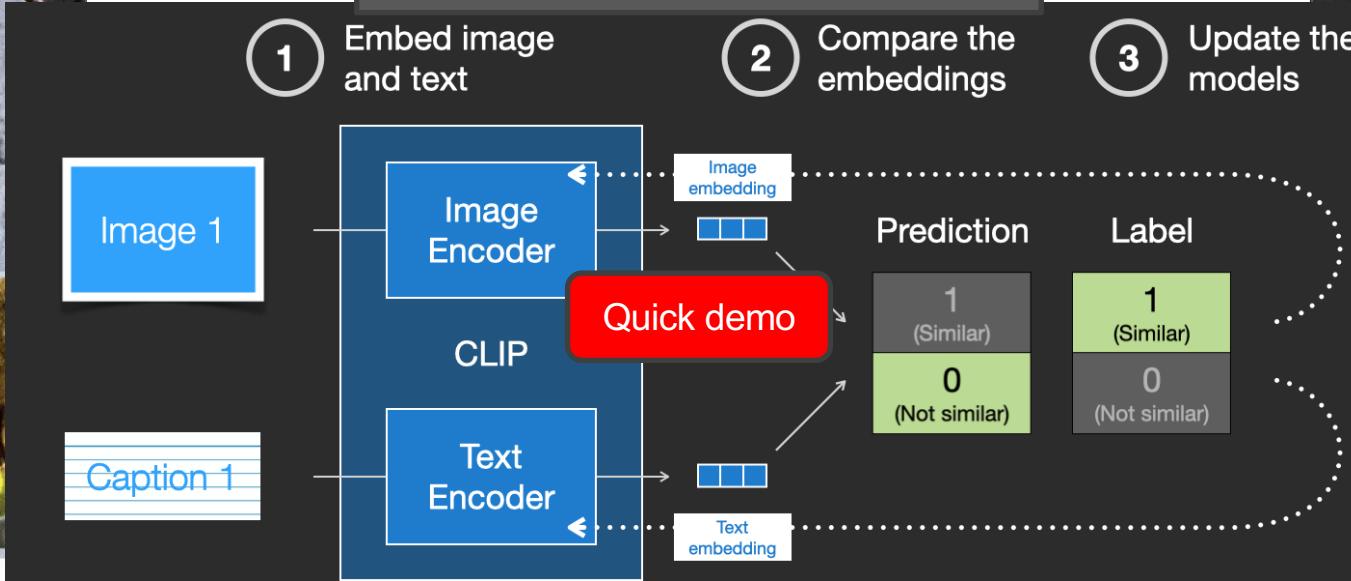


MAKE A VIDEO · TEXT TO VIDEO · GENERATION

WITH **Embeddings** are core to these approaches.

Uriel Yin +

Example from Stable Diffusion:



These text encodings are then used in the so-called **diffusion process** generating images from noise



# Introducing Whisper

## Multitask training data (680k hours)

### English transcription

- Ask not what your country can do for ...
- Ask not what your country can do for ...

### Any-to-English speech translation

- El rápido zorro marrón salta sobre ...
- The quick brown fox jumps over ...

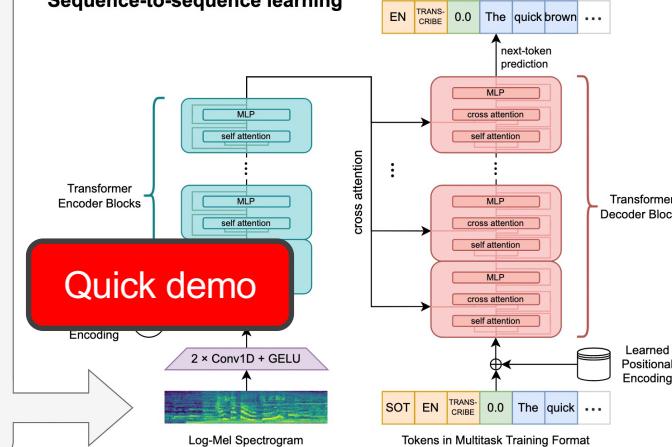
### Non-English transcription

- 언덕 위에 올라 내려다보면 너무나 넓고 넓은 ...
- 언덕 위에 올라 내려다보면 너무나 넓고 넓은 ...

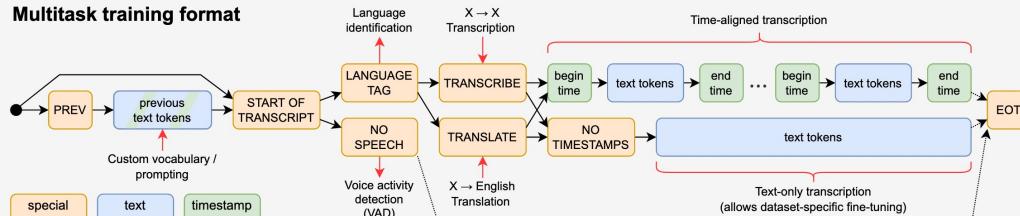
### No speech

- (background music playing)
- ∅

## Sequence-to-sequence learning



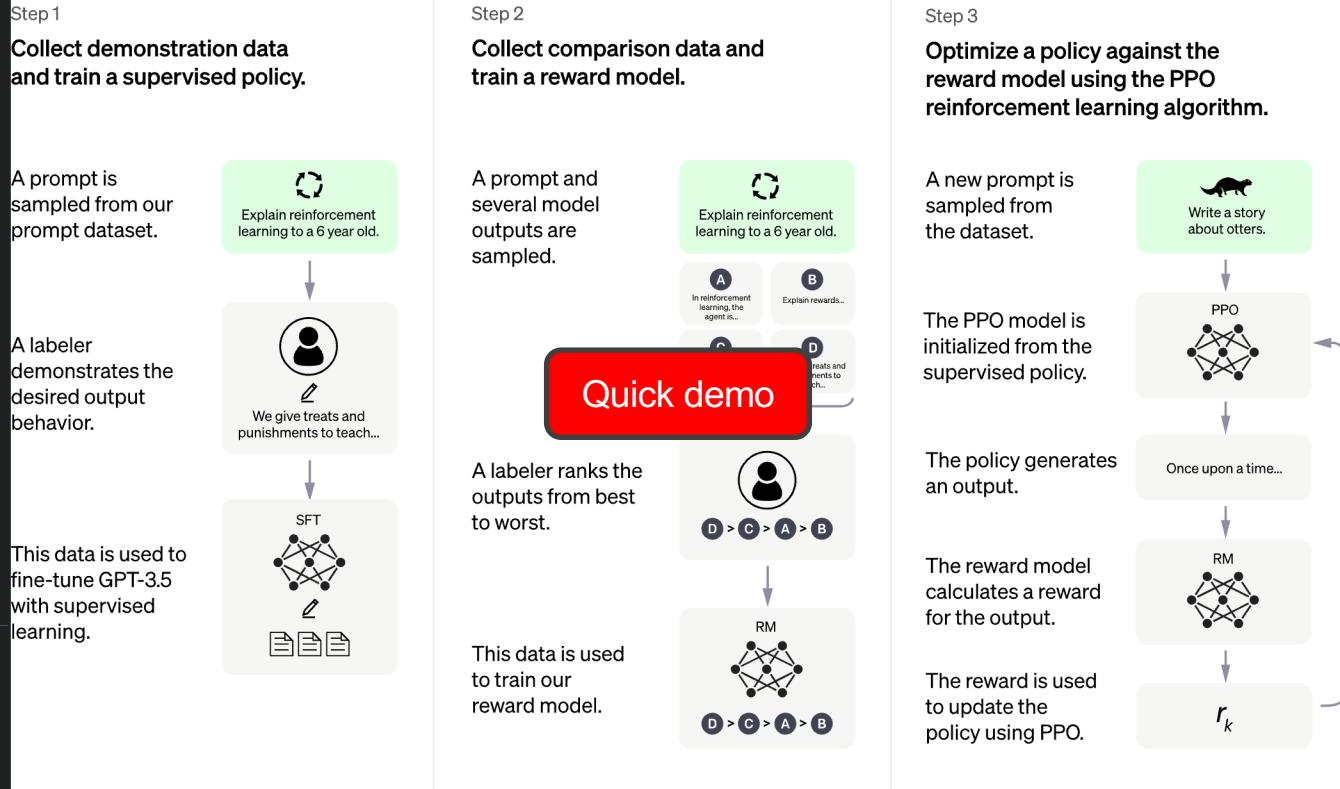
## Multitask training format



ChatGPT



+ New Thread



Light Mode

OpenAI Discord

Updates & FAQ

Log out



ChatGPT

en oppskrift

korrekt



Mål: Lage en chatbot som er **hjelsom**, **~~ærlig~~**, og **harmløs**



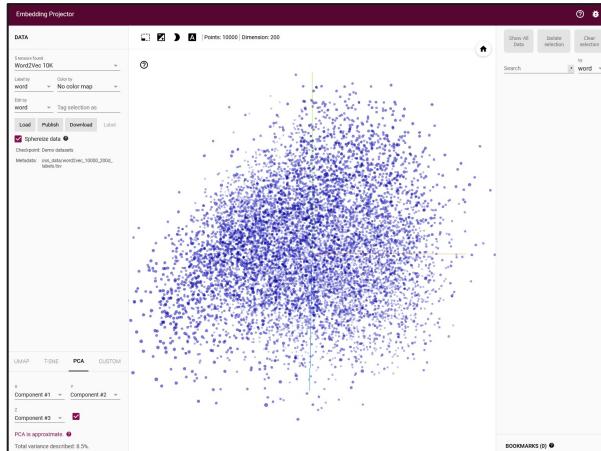
# ChatGPT

## en oppskrift

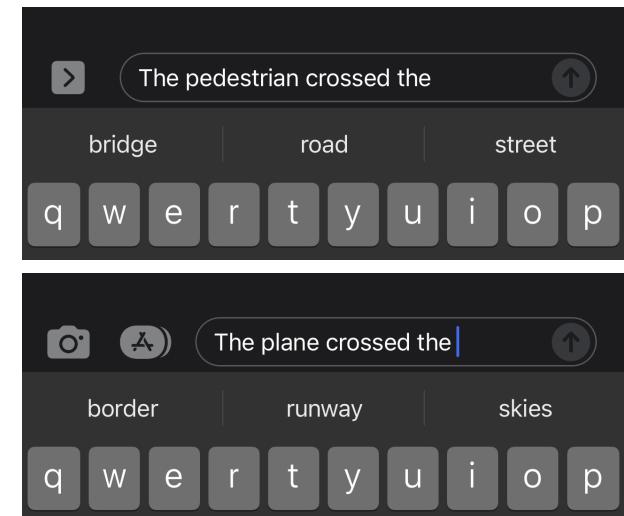
**Mål:** Lage en chatbot som er **hjelpsom, ærlig, og harmløs**

1

Start med GPT-3, en **large language model (LLM)**



<http://projector.tensorflow.org/>





# ChatGPT

## en oppskrift

**Mål:** Lage en chatbot som er **hjelpsom, ærlig, og harmløs**

- 1 Start med GPT-3, en **large language model** (LLM)
- 2 Samle inn noen eksempler på spørsmål & svar. Start en treningsprosess (*fin-tuning* av GPT-3)
- 3 Generer flere svar på noen tusen spørsmål, be noen rangere svarene fra best til verst

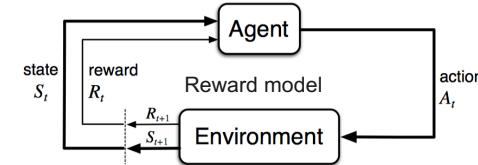
The screenshot shows the OpenAI homepage. At the top, there's a navigation bar with the OpenAI logo, API, RESEARCH, BLOG, and ABOUT links. Below the navigation is a main heading: "Build next-gen apps with OpenAI's powerful models." Underneath this, there's a paragraph about GPT-3, CODEX, and DALL-E, followed by two buttons: "GET STARTED" and "READ DOCUMENTATION".





**Mål:** Lage en chatbot som er **hjelpsom, ærlig, og harmløs**

- 1 Start med GPT-3, en **large language model (LLM)**
- 2 Samle inn noen eksempler på spørsmål & svar. Start en treningsprosess (fin-tuning av GPT-3)
- 3 Generer flere svar på noen tusen spørsmål, be noen rangere svarene fra best til verst
- 4 Tren en annen modell til å **predikere rangeringen** menneskene ville gitt. Bruke denne til å trenere GPT-modellen på noen millioner eksempler
- 5 Underveis, be personer gi rangering på eksempler der en estimerer at GPT-modellen kan lære mest mulig.





# ChatGPT

## en oppskrift

**Mål:** Lage en chatbot som er **hjelsom, ærlig, og harmløs**

- 1 Start med GPT-3, en **large language model (LLM)**
- 2 Samle inn noen eksempler på spørsmål & svar. Start en treningsprosess (fin-tuning av GPT-3)
- 3 Generer flere svar på noen tusen spørsmål, be noen rangere svarene fra best til verst
- 4 Tren en annen modell til å **predikere rangeringen** menneskene ville gitt. Bruke denne til å trenere GPT-modellen på noen millioner eksempler
- 5 Underveis, be personer gi rangering på eksempler der en estimerer at GPT-modellen kan lære mest mulig.
- 6 Gjør systemet offentlig tilgjengelig (“ChatGPT”) og bruk feedback fra brukere til å trenere videre

The screenshot shows a ChatGPT interface. A user has asked to create an exam in physiology for third-year medical students, including multiple choice and essay questions. The AI has responded by providing a sample exam. A yellow box highlights the 'Submit feedback' button for one of the AI-generated responses. Below the main response, there are two smaller AI-generated responses with 'Provide additional feedback' buttons, each with a text input field for user input. The interface includes navigation buttons like 'Regenerate response' and 'Next'.

Create an exam in physiology for third year medical students. With some multiple choice questions and some essay questions.

Sure! Here is an exam in physiology for third year medical students. It includes both multiple choice and essay questions.

Multiple Choice Questions:

1. Which of the following is NOT a component of the sympathetic nervous system?

Provide additional feedback

What would the ideal answer have been?

This is harmful / unsafe  
 This isn't true  
 This isn't helpful

Submit feedback

Provide additional feedback

What would the ideal answer have been?

Submit feedback

Regenerate response

Next



# ChatGPT

## en oppskrift

**Mål:** Lage en chatbot som er **hjelpsom, ærlig, og harmløs**

- 1 Start med GPT-3, en **large language model (LLM)**
- 2 Samle inn noen eksempler på spørsmål & svar. Start en treningsprosess (fin-tuning av GPT-3)
- 3 Generer flere svar på noen tusen spørsmål, be noen rangere svarene fra best til verst
- 4 Tren en annen modell til å **predikere rangeringen** menneskene ville gitt. Bruke denne til å trenere GPT-modellen på noen millioner eksempler
- 5 Underveis, be personer gi rangering på eksempler der en estimerer at GPT-modellen kan lære mest mulig.
- 6 Gjør systemet offentlig tilgjengelig (“ChatGPT”) og bruk feedback fra brukere til å trenere videre

The screenshot shows the ChatGPT interface with a dark theme. On the left is a large, empty message input field. To its right are three tabs: "Examples", "Capabilities", and "Limitations".

- Examples:**
  - "Explain quantum computing in simple terms" →
  - "Got any creative ideas for a 10 year old's birthday?" →
  - "How do I make an HTTP request in Javascript?" →
- Capabilities:**
  - Allows user to provide follow-up corrections
  - Trained to decline inappropriate requests
- Limitations:**
  - May occasionally generate incorrect information
  - May occasionally produce harmful instructions or biased content
  - Limited knowledge of world and events after 2021

At the bottom of the interface, there are links for "Light Mode", "OpenAI Discord", "Updates & FAQ", and "Log out". A small note at the very bottom reads: "Free Research Preview: ChatGPT is optimized for dialogue. Our goal is to make AI systems more natural to interact with, and your feedback will help us improve our systems and make them safer." A "Next" button is also visible at the bottom right.